```
from sklearn.datasets import fetch_openml

X, y = fetch_openml('mnist_784', version=1, return_X_y=True)

#rescale
X = X / 255.

X_train, X_test = X[:60000], X[60000:]
y_train, y_test = y[:60000], y[60000:]


import numpy as np
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

pca = PCA()


X_train.shape
```

```
(60000, 784)
```

```
print(X_train[0])
```

```
[0.         0.         0.         0.         0.         0.19215686
 0.93333333 0.99215686 0.99215686 0.99215686 0.99215686 0.99215686
 0.99215686 0.99215686 0.99215686 0.98431373 0.36470588 0.32156863
 0.32156863 0.21960784 0.15294118 0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.07058824 0.85882353 0.99215686
 0.99215686 0.99215686 0.99215686 0.99215686 0.77647059 0.71372549
 0.96862745 0.94509804 0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.31372549 0.61176471 0.41960784 0.99215686
 0.99215686 0.80392157 0.04313725 0.         0.16862745 0.60392157
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.05490196 0.00392157 0.60392157 0.99215686 0.35294118
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.54509804 0.99215686 0.74509804 0.00784314 0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.04313725
 0.74509804 0.99215686 0.2745098  0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.1372549  0.94509804
 0.88235294 0.62745098 0.42352941 0.00392157 0.         0.
 0.         0.         0.         0.         0.         0.
 0.         0.         0.         0.         0.         0.
```

```
       0.         0.         0.         0.         0.         0.
       0.         0.         0.         0.31764706 0.94117647 0.99215686
       0.99215686 0.46666667 0.09803922 0.         0.         0.
       0.         0.         0.         0.         0.         0.
       0.         0.         0.         0.         0.         0.
       0.         0.         0.         0.         0.         0.
       0.         0.         0.17647059 0.72941176 0.99215686 0.99215686
       0.58823529 0.10588235 0.         0.         0.         0.
       0.         0.         0.         0.         0.         0.
       0.         0.         0.         0.         0.         0.
       0.         0.         0.         0.         0.         0.
       0.         0.0627451  0.36470588 0.98823529 0.99215686 0.73333333
       0.         0.         0.         0.         0.         0.
       0.         0.         0.         0.         0.         0.
       0.         0.         0.         0.         0.         0.
       0.         0.         0.         0.         0.         0.
       0.         0.97647059 0.99215686 0.97647059 0.25098039 0.
       0.         0.         0.         0.         0.         0.
       0.         0.         0.         0.         0.         0.
       0.         0.         0.         0.         0.         0.
       0.         0.         0.18039216 0.50980392 0.71764706 0.99215686
       0.99215686 0.81176471 0.00784314 0.         0.         0.
       0.         0.         0.         0.         0.         0.
       0.         0.         0.         0.         0.         0.
       0.         0.         0.         0.         0.15294118 0.58039216
       0.89803922 0.99215686 0.99215686 0.99215686 0.98039216 0.71372549
       0.         0.         0.         0.         0.         0.
```

```
n_comp = 30

pca = PCA(n_components=n_comp)

pca.fit(X_train)
#30次元の潜在空間に変換して圧縮
X_train_latent = pca.transform(X_train)
```

```
X_train_latent.shape
```

```
(60000, 30)
```

```
X_train_latent[0]
```

```
array([ 0.48601015, -1.22617358, -0.09613354, -2.17944298, -0.10704577,
       -0.91167171,  0.91763043,  0.62666548, -1.4255497 ,  0.77815136,
        0.77449643, -0.99630654, -0.44509738,  2.93841289,  0.85980051,
       -0.01848352,  1.29452532,  1.21255137,  1.08903686,  0.65102536,
        0.10739394, -0.25150383, -0.8499285 ,  0.98205115,  0.1859757 ,
        0.40418704, -1.22074111,  0.65974312, -0.49827378, -0.55485382])
```

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.font_manager
from sklearn import svm

# fit the model
clf = svm.OneClassSVM(nu=0.2, kernel='rbf', gamma=0.1)
clf.fit(X_train_latent)
```

```
      OneClassSVM(cache_size=200, coef0=0.0, degree=3, gamma=0.1, kernel='rbf',
                  max_iter=-1, nu=0.2, shrinking=True, tol=0.001, verbose=False)
```

```
y_pred_train_latent = clf.predict(X_train_latent)
```

```
print(y_pred_train_latent)
```

```
      [ 1  1  1 ... -1  1  1]
```

```
y_pred_train_latent.shape
```

```
      (60000,)
```

```
y_pred_train_latent[0]
```

```
      1
```

```
extension_2 =[]
for i in range(10):
    if i%2==0:
        extension_2.append(i)
extension_2
#>>> [0, 2, 4, 6, 8]
```

```
      [0, 2, 4, 6, 8]
```

```
kara = []
for i in range(100):
   if y_pred_train_latent[i]==-1:
              kara.append(i)
kara
```

```
      [27, 28, 31, 34, 47, 51, 56, 60, 63, 64, 70, 75, 82, 83]
```

```
len(kara)
```

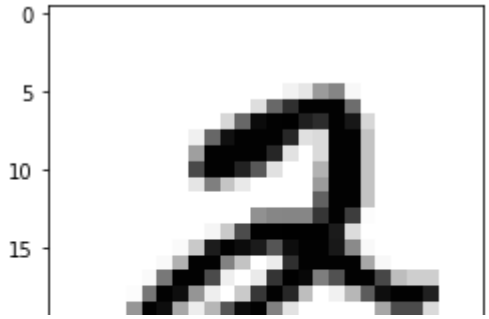
```
      14
```

```
kara[0]
```

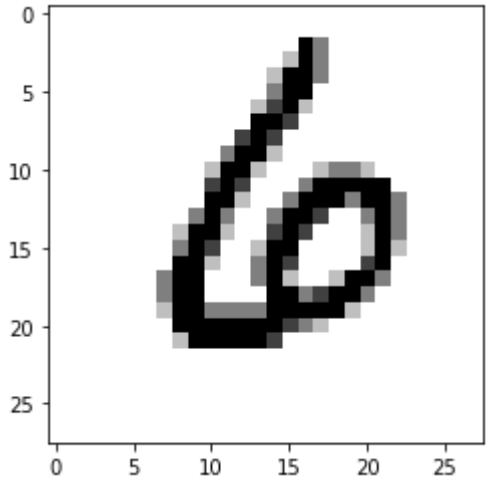
```
      27
```

```
plt.imshow(X[5].reshape(28,28), cmap=plt.cm.gray_r)
```

```
<matplotlib.image.AxesImage at 0x7f5554fde610>
```



```
for i in range(len(kara)):
    plt.imshow(X[kara[i]].reshape(28,28), cmap=plt.cm.gray_r)
```



```
import numpy as np
import cv2
```

```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Mounted at /content/gdrive
```

```
%cd ./gdrive/'My Drive'/"Colab Notebooks"/"extractdouga"
```

```
/content/gdrive/My Drive/Colab Notebooks/extractdouga
```

```
cv2.imwrite('henkantest.jpg',255**X[kara[0]].reshape(28,28,1))
```

```
True
```

```
X[kara[0]].reshape(28,28,1)
```

```
       [0.        ]],

      [[0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ],
       [0.        ]
```

```
       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ]],

      [[0.         ],
       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ],

       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ],
       [0.         ]]])
```

```
for i in range(len(kara)):
  cv2.imwrite(f'comb{i:03d}.jpg',255**X[kara[i]].reshape(28,28,1))
```

#ファイルに問題あり

```
import sys
import cv2
```

```
# encoder(for mp4)
fourcc = cv2.VideoWriter_fourcc('m', 'p', '4', 'v')
# output file name, encoder, fps, size(fit to image size)
video = cv2.VideoWriter('video.mp4',fourcc, 20.0, (1240, 1360))

if not video.isOpened():
    print("can't be opened")
    sys.exit()

for i in range(0,len(kara) ):
    # hoge0000.png, hoge0001.png,..., hoge0090.png
    img = cv2.imread('./comb%03d.jpg' % i)

    # can't read image, escape
    if img is None:
        print("can't read")
        break

    # add
    video.write(img)
    print(i)

video.release()
print('written')
```

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
written
```

```
import glob

from PIL import Image

frames = []
images = sorted(glob.glob("./*.jpg"))

print(images)
```

```
['./comb000.jpg', './comb001.jpg', './comb002.jpg', './comb003.jpg', './comb004.jpg', './comb
```

```
for image in images:
```

```
        new_frame = Image.open(image)
        frames.append(new_frame)

    frames[0].save('jpg_to_gif.gif',
                   format='GIF',
                   append_images=frames[1:],
                   save_all=True,
                   duration=500,
                   loop=0)
```

✓ 0 秒　完了時間: 13:46